# **SED-505**

# **ASSIGNMENT-2 TRAVEL SIMULATION**

**October 5, 2025**

**MIGUEL WATLER**

**MICHAEL MELLES 111838223**

**QUESTIONS:**

**1. What type of design pattern did you employ in this assignment? Why?**

I applied a structural design pattern—the Adapter Pattern. The Adapter Pattern bridges the gap between two incompatible interfaces so that legacy classes (Boat, Car, and Airplane) can be used seamlessly under a common simulation system. By wrapping these classes with adapter classes that specify a uniform interface (IVehicle), I was able to normalize their behavior without modifying their original implementations. This approach prefers reusability of the code, separation of concerns, and flexibility to introduce new vehicle types into the system.

**2. Describe a software design pattern that you have used before this course, even if you were not aware of it.**

Before this course, I unintentionally used the Observer Pattern in implementing a simple notification system in a to-do list app. In the project, when a task was updated, multiple components (like the UI, database logger, and email notifier) needed to react to the update. I addressed this by registering each component to listen for task updates and behave accordingly oblivious that I was using a well-known behavioral pattern. The Observer Pattern is ideal for such scenarios where one change must propagate through multiple listeners.

**3. Do you feel standard design patterns are of great assistance, or great hindrance, in software design? Give advantages and disadvantages.**

I believe standard design patterns are a great assistance when used thoughtfully. They offer proven solutions to common problems and help teams communicate ideas more clearly. However, they can become a hindrance if applied rigidly or without understanding.

*Advantages:*

1. Improved code readability – Patterns provide a shared vocabulary for developers.
2. Reusability – They encourage modular and maintainable code.
3. Scalability – Patterns like Adapter and Strategy make systems easier to extend.
4. Educational value – They help developers learn best practices and architectural thinking.

*Disadvantages:*

1. Overengineering – Using patterns where simple solutions would suffice can complicate code.
2. Rigid structures – Some patterns can introduce unnecessary constraints.
3. Misuse by beginners – Without proper understanding, patterns can be misapplied and lead to poor design.

**4. What are some major drawbacks of the Singleton design pattern?**

1. Global State Risk Singletons often behave like global variables, which can lead to hidden dependencies and make debugging difficult.
2. Testing Challenges Since Singletons maintain state, they can interfere with unit tests and make mocking or resetting behavior harder.
3. Concurrency Issues In multi-threaded environments, improper Singleton implementation can lead to race conditions or synchronization problems.
4. Tight Coupling Classes that depend on a Singleton are tightly coupled to its implementation, reducing flexibility and violating the Dependency Inversion Principle.
5. Hidden Dependencies Because Singletons are accessed globally, it is easy to overlook where and how they are used, making the codebase harder to maintain